Real Estate Predictions

Thomas McGiverin

Introduction

In this project, I will be building machine learning models for the purpose of predicting the price of a house given a set of features related to the house.

The dataset I will be using is the Ames Housing dataset. The data describes the sale of individual residential properties in Ames, Iowa from 2006 to 2010. There are 1460 observations with 79 explanatory variables (nominal, ordinal, discrete, and continuous) which are involved in the assessment of the value of a house.

Variable Selection

```
# Read in data
raw_housing_data <- read.csv("train.csv")

# Display names of columns
names(raw_housing_data)
```

```
##  [1] "Id"            "MSSubClass"    "MSZoning"      "LotFrontage"
##  [5] "LotArea"       "Street"        "Alley"         "LotShape"
##  [9] "LandContour"   "Utilities"     "LotConfig"     "LandSlope"
## [13] "Neighborhood"  "Condition1"    "Condition2"    "BldgType"
## [17] "HouseStyle"    "OverallQual"   "OverallCond"   "YearBuilt"
## [21] "YearRemodAdd"  "RoofStyle"     "RoofMatl"      "Exterior1st"
## [25] "Exterior2nd"   "MasVnrType"    "MasVnrArea"    "ExterQual"
## [29] "ExterCond"     "Foundation"    "BsmtQual"      "BsmtCond"
## [33] "BsmtExposure"  "BsmtFinType1"  "BsmtFinSF1"    "BsmtFinType2"
## [37] "BsmtFinSF2"    "BsmtUnfSF"     "TotalBsmtSF"   "Heating"
## [41] "HeatingQC"     "CentralAir"    "Electrical"    "X1stFlrSF"
## [45] "X2ndFlrSF"     "LowQualFinSF"  "GrLivArea"     "BsmtFullBath"
## [49] "BsmtHalfBath"  "FullBath"      "HalfBath"      "BedroomAbvGr"
## [53] "KitchenAbvGr"  "KitchenQual"   "TotRmsAbvGrd"  "Functional"
## [57] "Fireplaces"    "FireplaceQu"   "GarageType"    "GarageYrBlt"
## [61] "GarageFinish"  "GarageCars"    "GarageArea"    "GarageQual"
## [65] "GarageCond"    "PavedDrive"    "WoodDeckSF"    "OpenPorchSF"
## [69] "EnclosedPorch" "X3SsnPorch"    "ScreenPorch"   "PoolArea"
## [73] "PoolQC"        "Fence"         "MiscFeature"   "MiscVal"
## [77] "MoSold"        "YrSold"        "SaleType"      "SaleCondition"
## [81] "SalePrice"
```

With such a large number of features to include in a model, I will be selecting a subset of features which I believe to be strong predictors in the value of a house. The features I will choose come from my knowledge of the real estate market and what I know to have an impact on the house price.

It is important to consider that there are only 1460 observation in the test set and many of the categorical

variables have a large number of levels which can make the impractical to use.

```
raw_subset <- raw_housing_data %>% select(BldgType, LotArea, Street, Utilities,
                                          HouseStyle, OverallQual, OverallCond,
                                          YearBuilt, YearRemodAdd, BsmtFinType1,
                                          BsmtUnfSF, Heating, CentralAir,
                                          GrLivArea, FullBath, BedroomAbvGr,
                                          GarageCars, SalePrice)
```

- BldgType: Type of dwelling
- LotArea: Lot size in square feet
- Street: Type of road access to property
- Utilities: Type of utilities available
- HouseStyle: Style of dwelling
- OverallQual: Rates the overall material and finish of the house
- OverallCond: Rates the overall condition of the house
- YearBuilt: Original construction date
- YearRemodAdd: Remodel date (same as construction date if no remodeling or additions)
- BsmtFinType1: Rating of basement finished area
- BsmtUnfSF: Unfinished square feet of basement area
- Heating: Type of heating
- CentralAir: Central air conditioning
- GrLivArea: Above grade (ground) living area square feet
- FullBath: Full bathrooms above grade
- BedroomAbvGr: Bedrooms above grade (does NOT include basement bedrooms)
- GarageCars: Size of garage in car capacity

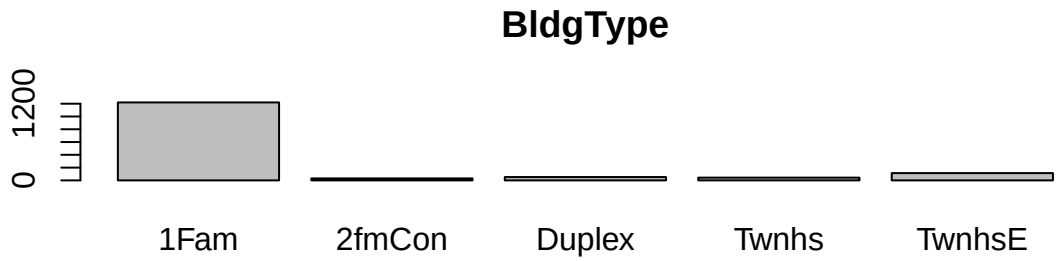
This is the list of variables I have chosen to be most impactful from the full list of 79. I have been able to narrow down the number of features to 17, however, for such a small sample size I believe that less features would still be better. To narrow down the list of features more, I will perform EDA on the 17 features to determine if any of them seem to carry little to no information and can then be removed from the analysis.

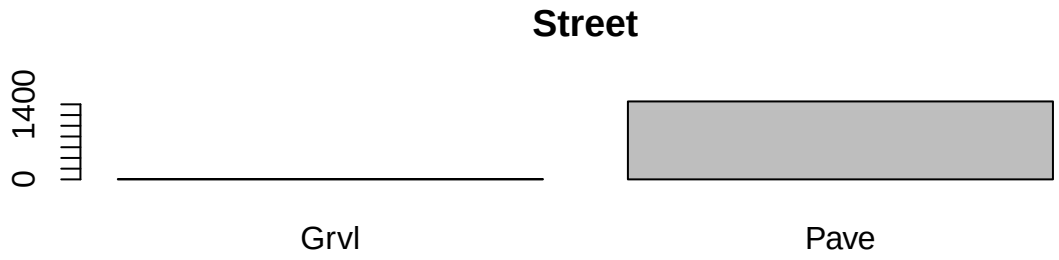
Feature Analysis

Categorical variables

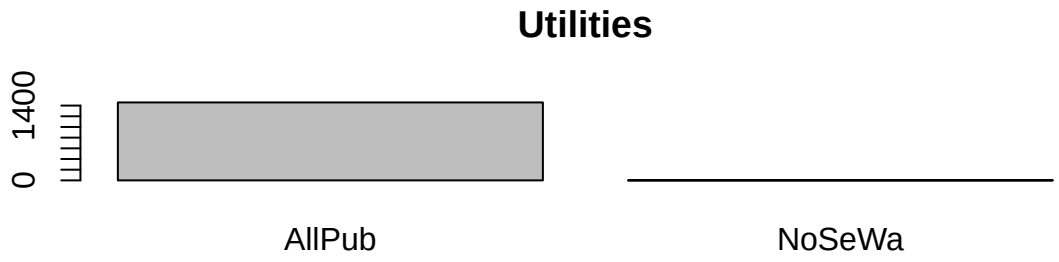
Here I will analyze all categorical features to determine if any can be removed.

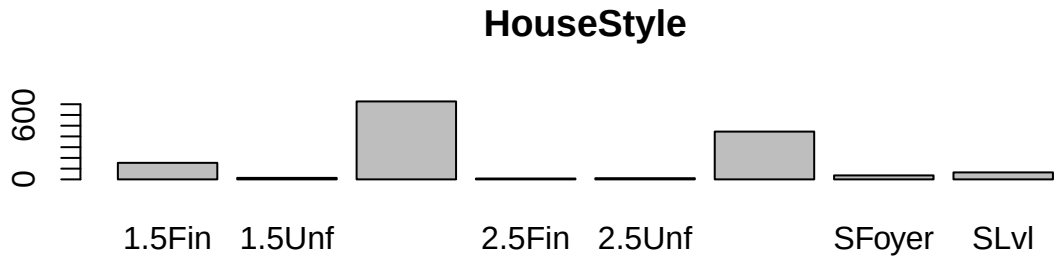
```
# Convert character columns to factors
raw_subset$BldgType <- as.factor(raw_subset$BldgType)
raw_subset$Street <- as.factor(raw_subset$Street)
raw_subset$Utilities <- as.factor(raw_subset$Utilities)
raw_subset$HouseStyle <- as.factor(raw_subset$HouseStyle)
raw_subset$BsmtFinType1 <- as.factor(raw_subset$BsmtFinType1)
raw_subset$Heating <- as.factor(raw_subset$Heating)
raw_subset$CentralAir <- as.factor(raw_subset$CentralAir)

# Create separate dataframe for categorical variables
raw_cat <- raw_subset %>% select(BldgType, Street, Utilities, HouseStyle,
                                 BsmtFinType1, Heating, CentralAir, SalePrice)

# Plot a histogram for each of the categorical variables to analyze distribution
par(mfrow=c(2,1))
plot(raw_cat$BldgType, main="BldgType")
plot(raw_cat$Street, main="Street")
```



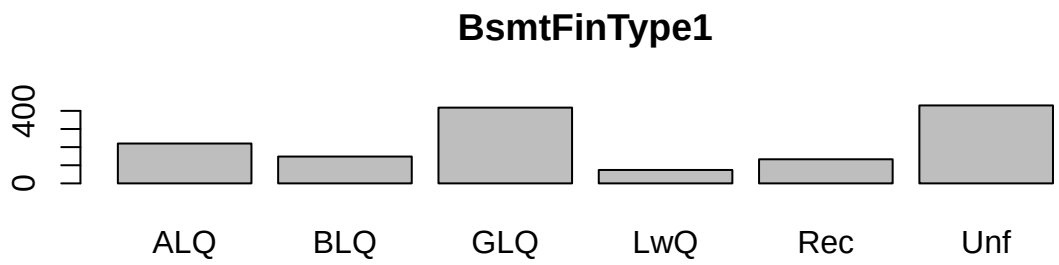


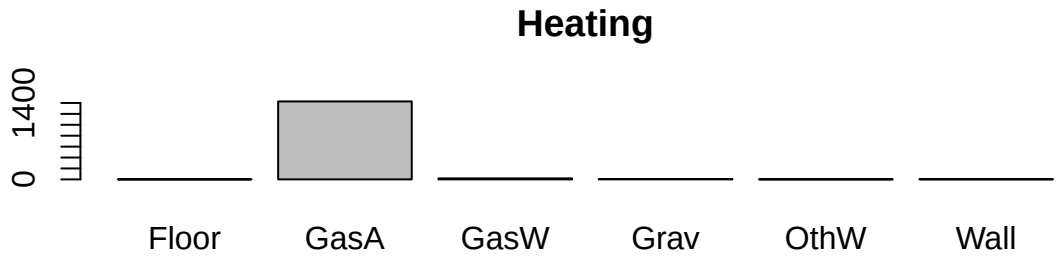

```
par(mfrow=c(2,1))
plot(raw_cat$Utilities, main="Utilities")
plot(raw_cat$HouseStyle, main="HouseStyle")
```



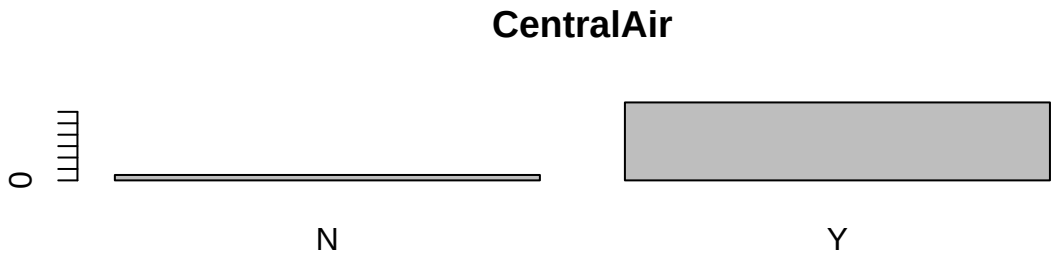



```
par(mfrow=c(2,1))
plot(raw_cat$BsmtFinType1, main="BsmtFinType1")
plot(raw_cat$Heating, main="Heating")
```

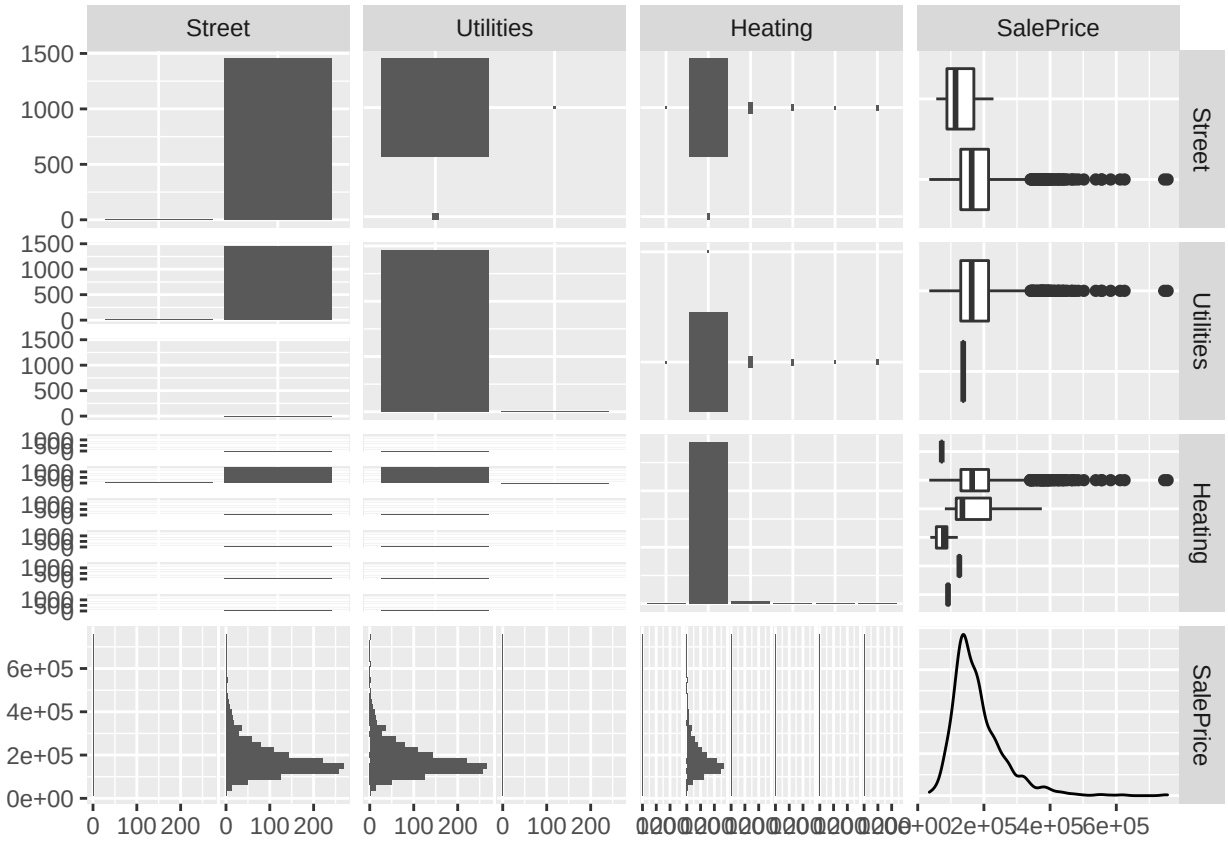





```
par(mfrow=c(2,1))
plot(raw_cat$CentralAir, main="CentralAir")


# Create a pairs plot to analyze the variables which are heavily imbalanced
par(mfrow=c(1,1))
```




```
ggpairs(raw_cat[c("Street", "Utilities", "Heating", "SalePrice")])
```




```
# Summary of categorical variables to determine the distributions numerically
summary(raw_cat)
```

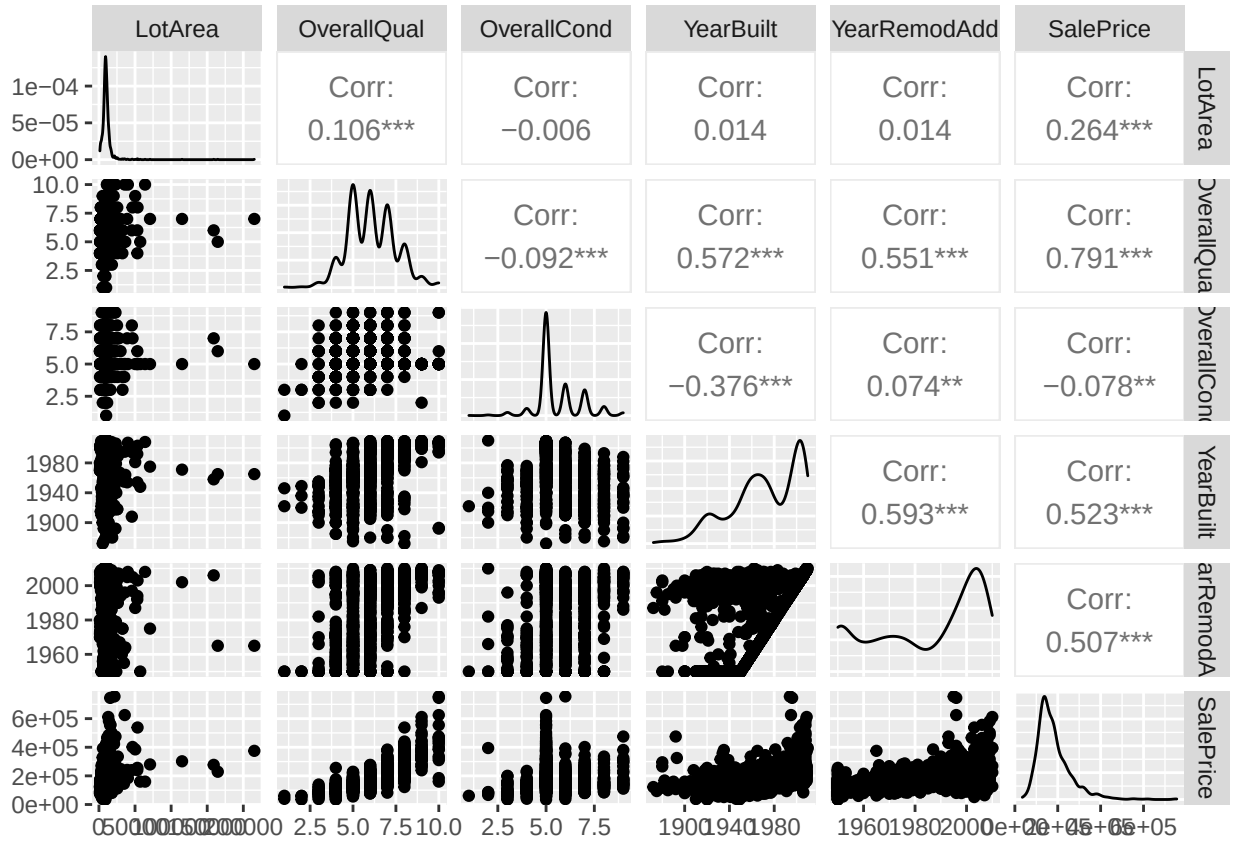
```
##    BldgType     Street      Utilities      HouseStyle  BsmtFinType1
##  1Fam  :1220   Grvl:   6   AllPub:1459   1Story :726   ALQ :220
##  2fmCon:  31   Pave:1454   NoSeWa:   1   2Story :445   BLQ :148
##  Duplex:  52                             1.5Fin :154   GLQ :418
##  Twnhs :  43                             SLvl   : 65   LwQ : 74
##  TwnhsE: 114                             SFoyer : 37   Rec :133
##                                          1.5Unf : 14   Unf :430
##                                          (Other): 19   NA's: 37
##   Heating     CentralAir   SalePrice
##  Floor:   1   N:  95      Min.   : 34900
##  GasA :1428   Y:1365      1st Qu.:129975
##  GasW :  18               Median :163000
##  Grav :   7               Mean   :180921
##  OthW :   2               3rd Qu.:214000
##  Wall :   4               Max.   :755000
##
```

From these charts it appears that we can exclude Street, Utilities, and Heating since their levels are very imbalanced with one level having nearly all observations.

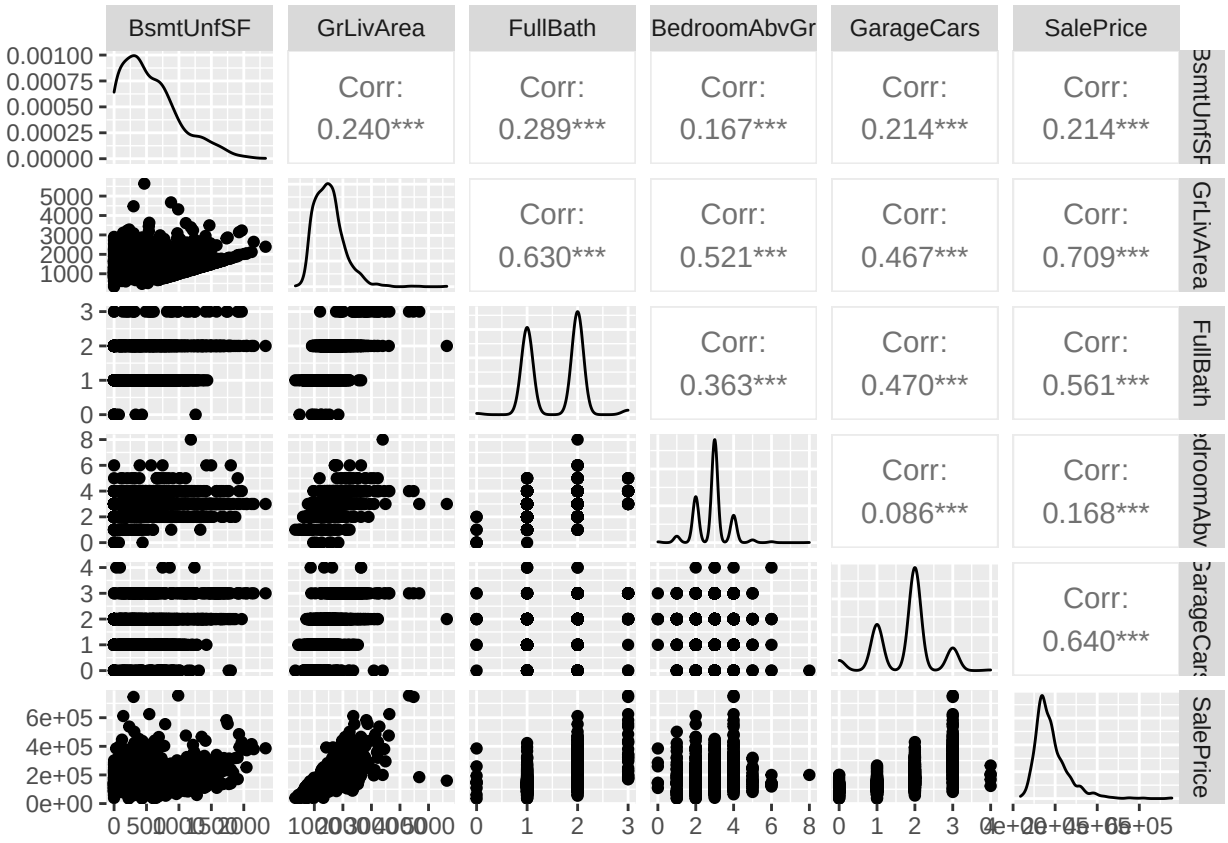
Numerical variables

I will now perform an analysis for all numerical variables to determine if any can be removed to increase the simplicity of future models.

```r
# Create dataframe for numerical features
raw_num <- raw_subset %>% select(LotArea, OverallQual, OverallCond, YearBuilt,
                                 YearRemodAdd, BsmtUnfSF, GrLivArea, FullBath,
                                 BedroomAbvGr, GarageCars, SalePrice)

# Create pairs plot for all numerical variables
ggpairs(raw_num[c("LotArea", "OverallQual", "OverallCond", "YearBuilt",
                  "YearRemodAdd", "SalePrice")])
```




```r
ggpairs(raw_num[c("BsmtUnfSF", "GrLivArea", "FullBath", "BedroomAbvGr",
                  "GarageCars", "SalePrice")])
```



Based on the analysis of these plots, I am going to remove YearRemodAdd since this feature is duplicating a lot of information from YearBuilt and the correlation between sales price and YearRemodAdd and YearBuilt is very similar.

Final Selection

Based on the above analysis of the 17 features that I had initially selected as being impactful to the value of a house, I have removed 3 categorical features and 1 numerical feature. These 13 features will be used moving forwards for the prediction of the value of the price of a house.

- BldgType: Type of dwelling
- LotArea: Lot size in square feet
- HouseStyle: Style of dwelling
- OverallQual: Rates the overall material and finish of the house
- OverallCond: Rates the overall condition of the house
- YearBuilt: Original construction date
- BsmtFinType1: Rating of basement finished area
- BsmtUnfSF: Unfinished square feet of basement area
- CentralAir: Central air conditioning
- GrLivArea: Above grade (ground) living area square feet
- FullBath: Full bathrooms above grade
- BedroomAbvGr: Bedrooms above grade (does NOT include basement bedrooms)
- GarageCars: Size of garage in car capacity

```
# Create dataframe for the 13 features that have been selected
house_subset <- raw_housing_data %>% select(Id, BldgType, LotArea, HouseStyle,
                                            OverallQual, OverallCond, YearBuilt,
```

```
                                        BsmtFinType1, BsmtUnfSF, CentralAir,
                                        GrLivArea, FullBath, BedroomAbvGr,
                                        GarageCars, SalePrice)


# Output head of data
kable(head(house_subset),
      col.names = c("Id", "Bldg", "Area", "Style", "Qual", "Cond", "Built", "BsmtType",
                    "BsmtSF", "AC", "LivArea", "Bath", "Bed", "Garage", "Price"))
```

Id	Bldg	Area	Style	Qual	Cond	Built	BsmtType	BsmtSF	AC	LivArea	Bath	Bed	Garage	Pri
1	1Fam	8450	2Story	7	5	2003	GLQ	150	Y	1710	2	3	2	2085
2	1Fam	9600	1Story	6	8	1976	ALQ	284	Y	1262	2	3	2	1815
3	1Fam	11250	2Story	7	5	2001	GLQ	434	Y	1786	2	3	2	2235
4	1Fam	9550	2Story	7	5	1915	ALQ	540	Y	1717	1	3	3	1400
5	1Fam	14260	2Story	8	5	2000	GLQ	490	Y	2198	2	4	3	2500
6	1Fam	14115	1.5Fin	5	5	1993	GLQ	64	Y	1362	1	1	2	1430

Preprocessing

Missing Data

In this section I will check the quality of the data and perform and necessary preprocessing that will be required for future analysis of the data.

First I will check if there are any missing values for the features.

```
# Count the number of NA values in each column
na_count <- sapply(house_subset, function(y) sum(length(which(is.na(y)))))
kable(data.frame(na_count))
```

	na_count
Id	0
BldgType	0
LotArea	0
HouseStyle	0
OverallQual	0
OverallCond	0
YearBuilt	0
BsmtFinType1	37
BsmtUnfSF	0
CentralAir	0
GrLivArea	0
FullBath	0
BedroomAbvGr	0
GarageCars	0
SalePrice	0

Here we see that BsmtFinType1 is the only column with missing values. Checking the data dictionary provided with the data file, it states that BsmtFinType1 is NA if the house does not have a basement. To remedy this I shall change the NA values to "NO" to indicate that the house does not have a basement.

```r
# Replace NA with "NO"
house_subset$BsmtFinType1[is.na(house_subset$BsmtFinType1)] <- "NO"
```

Outliers

Here I will check all columns for outliers using Tukey's definition of an outlier

```r
# Take a numeric vector as input and output the upper and lower bound of the
# outlier boundary based on Tukey's definition
far_outlier <- function(v) {
  v <- v[!is.na(v)]

  q1 <- quantile(v, 0.25, names = F)
  q3 <- quantile(v, 0.75, names = F)

  return(c(low = (q1 - 1.5 * (q3 - q1)) ,high = (q3 + 1.5 * (q3 - q1))))
}

# Takes a numeric vector as input and outputs
is_outlier <- function(v) {
  v <- v[!is.na(v)]
  cutoff <- far_outlier(v)
  contains_outlier <- (v > cutoff[2]) | (v < cutoff[1])

  if (sum(contains_outlier) > 0) {
    return(T)
  } else {
    return(F)
  }
}

numeric_tibble <- dplyr::select_if(house_subset, is.numeric)

for (col in names(numeric_tibble)) {
  if (is_outlier(numeric_tibble[col])) {

    outlier_range <- far_outlier(numeric_tibble[col])

    print(paste(
      "Non-Outlier range for", col, "is",
      round(outlier_range[1], 2), "to",
      round(outlier_range[2], 2)
    ))

    print(paste(nrow(filter(house_subset, house_subset[col] > outlier_range[2] |
                                house_subset[col] < outlier_range[1] )), "outliers found"))
  }
}
```

```
## [1] "Non-Outlier range for LotArea is 1481.5 to 17673.5"
## [1] "69 outliers found"
## [1] "Non-Outlier range for OverallQual is 2 to 10"
## [1] "2 outliers found"
## [1] "Non-Outlier range for OverallCond is 3.5 to 7.5"
```

```
## [1] "125 outliers found"
## [1] "Non-Outlier range for YearBuilt is 1885 to 2069"
## [1] "7 outliers found"
## [1] "Non-Outlier range for BsmtUnfSF is -654.5 to 1685.5"
## [1] "29 outliers found"
## [1] "Non-Outlier range for GrLivArea is 158.62 to 2747.62"
## [1] "31 outliers found"
## [1] "Non-Outlier range for BedroomAbvGr is 0.5 to 4.5"
## [1] "35 outliers found"
## [1] "Non-Outlier range for GarageCars is -0.5 to 3.5"
## [1] "5 outliers found"
## [1] "Non-Outlier range for SalePrice is 3937.5 to 340037.5"
## [1] "61 outliers found"
```

It is important to note here that outliers do not make sense for OverallQual and OverallCond since they are ordinal variables. Furthermore upon manual analysis of these outliers, I see no reason to believe that they are due to data entry errors nor do I have any reason to believe that they do not belong in the analysis. Therefore I will be including all observations which have been identified as outliers.If the outliers prove to be troublesome later on then I may consider removing them.

We can now proceed to using this preprocessed data to build statistical and machine learning models for the purpose of predicting the price of a house. I will now split the data into a train and test splits using 75% of the data for the train set and 25% for test.

```
# Split data into train and test set
set.seed(1000)

house_train <- house_subset %>% dplyr::sample_frac(.75)
house_test <- dplyr::anti_join(house_subset, house_train, by = "Id")

house_train <- house_train %>% select(-Id)
house_test <- house_test %>% select(-Id)
```

Satistical and Machine Learning Models for Prediction

Linear Regression

Linear regression is a great first model to try due to its simplicity and interpretability. If linear regression is a good model for the problem then a lot of the structure surrounding the relationships between the variables has been revealed.

Train

```
# Linear regression
house.lm <- lm(house_train$SalePrice~., data=house_train)

summary(house.lm)
```

```
##
## Call:
## lm(formula = house_train$SalePrice ~ ., data = house_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -419191  -17591   -1792   13161  262931
```

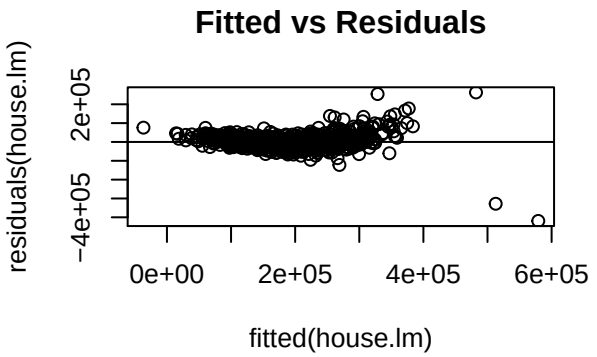
```
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -9.695e+05  1.290e+05  -7.516 1.19e-13 ***
## BldgType2fmCon   -2.442e+03  7.664e+03  -0.319 0.750027
## BldgTypeDuplex   -1.727e+04  6.724e+03  -2.568 0.010354 *
## BldgTypeTwnhs    -2.696e+04  6.831e+03  -3.947 8.43e-05 ***
## BldgTypeTwnhsE   -1.931e+04  4.495e+03  -4.296 1.90e-05 ***
## LotArea           4.264e-01  1.054e-01   4.047 5.56e-05 ***
## HouseStyle1.5Unf  9.834e+03  1.195e+04   0.823 0.410663
## HouseStyle1Story  1.642e+04  4.309e+03   3.809 0.000147 ***
## HouseStyle2.5Fin -1.326e+04  1.442e+04  -0.920 0.357997
## HouseStyle2.5Unf -1.933e+04  1.268e+04  -1.525 0.127615
## HouseStyle2Story -6.090e+03  4.416e+03  -1.379 0.168177
## HouseStyleSFoyer  6.341e+03  8.556e+03   0.741 0.458803
## HouseStyleSLvl   -3.009e+03  6.371e+03  -0.472 0.636799
## OverallQual       2.094e+04  1.398e+03  14.976  < 2e-16 ***
## OverallCond       5.863e+03  1.169e+03   5.015 6.22e-07 ***
## YearBuilt         4.480e+02  6.678e+01   6.708 3.19e-11 ***
## BsmtFinType1BLQ   2.494e+02  4.417e+03   0.056 0.954975
## BsmtFinType1GLQ   7.598e+03  3.840e+03   1.978 0.048129 *
## BsmtFinType1LwQ  -7.008e+03  5.749e+03  -1.219 0.223123
## BsmtFinType1NO   -1.334e+04  8.105e+03  -1.646 0.100023
## BsmtFinType1Rec  -3.832e+03  4.668e+03  -0.821 0.411953
## BsmtFinType1Unf  -6.676e+03  4.334e+03  -1.540 0.123778
## BsmtUnfSF        -7.616e+00  3.789e+00  -2.010 0.044702 *
## CentralAirY      -3.767e+03  5.269e+03  -0.715 0.474868
## GrLivArea         6.773e+01  3.884e+00  17.439  < 2e-16 ***
## FullBath          9.358e+02  3.066e+03   0.305 0.760271
## BedroomAbvGr     -6.211e+03  1.867e+03  -3.328 0.000906 ***
## GarageCars        1.182e+04  1.994e+03   5.927 4.16e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 35370 on 1067 degrees of freedom
## Multiple R-squared:  0.7969, Adjusted R-squared:  0.7917
## F-statistic:   155 on 27 and 1067 DF,  p-value: < 2.2e-16
```

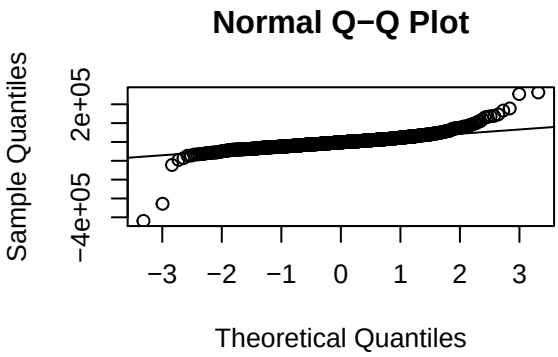
With an R-squared value of 0.7917, our model is able to explain 79.17% of the variability in the data which is a great start. Our F-statisitc is statistically significant which means a relationship between the house features and the house price does exist. Looking at the various t-tests, we can see that BsmtUnfSF (Unfinished basement square feet), CentralAirY (Air conditioning), FullBath (Number of bathrooms) are not statistically significant for this model.

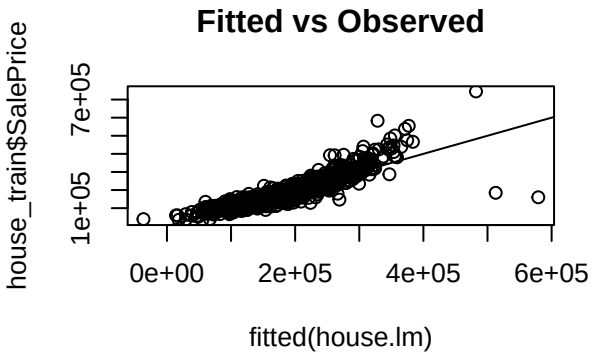
However, all of what I have mentioned above does not matter unless the assumptions underlying the linear regression hold. To check this I will plot the fitted vs residuals, a normal QQ plot, and the fitted vs observed values.

```r
#Diagnostic plots for the linear regression
par(mfrow=c(2,2))
plot(fitted(house.lm), residuals(house.lm), main="Fitted vs Residuals")
abline(h=0)

qqnorm(residuals(house.lm))
qqline(residuals(house.lm))
```

```
plot(fitted(house.lm), house_train$SalePrice, main="Fitted vs Observed")
abline(0,1)
```



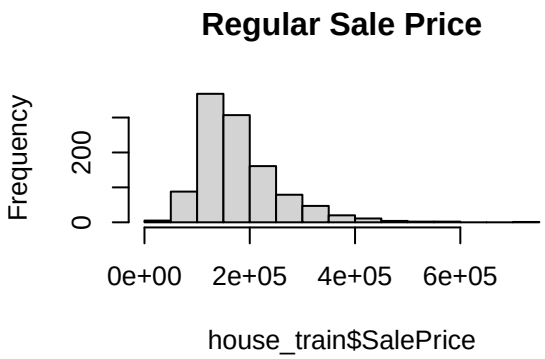


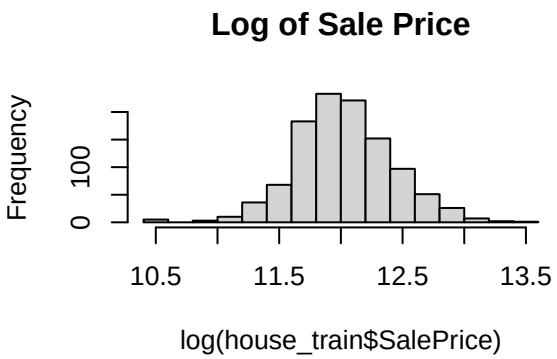


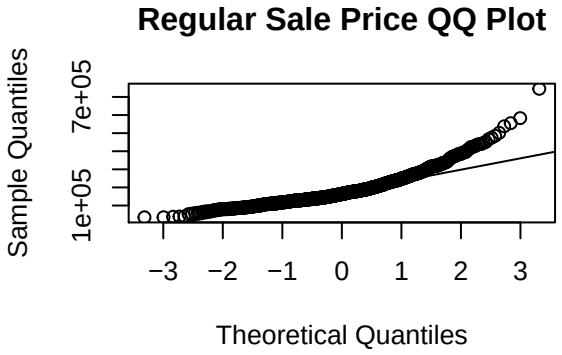
There is a fairly clear non-linear pattern in the fitted vs residuals plot which is a red flag. The normal QQ plot strays far from normality in the tails of the distribution which is a potential violation of normality. In the fitted vs observed values we see a non-linear pattern in this data. All of the diagnostics together point to a linear model not being very appropriate here and we should consider a non-linear model.

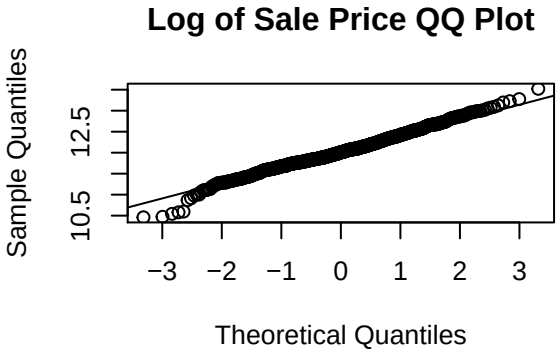
A very important fact to note is that the sale price of a house is a positively skewed distribution for this dataset. This leads to the natural application of the log transformation to sale price to bring the skewed distribution as close to normality as possible.

```
#Histograms of sale price vs log sale price
par(mfrow=c(2,2))
hist(house_train$SalePrice, main="Regular Sale Price")
hist(log(house_train$SalePrice), main="Log of Sale Price")

#Normal qq plots of sale price vs log sale price
qqnorm(house_train$SalePrice, main="Regular Sale Price QQ Plot")
qqline(house_train$SalePrice)

qqnorm(log(house_train$SalePrice), main="Log of Sale Price QQ Plot")
qqline(log(house_train$SalePrice))
```









We see that taking the log of the sale price brings the distribution very close to normality. With this information I shall perform another linear regression but with the log transformed sale price so that the assumptions of our model will be valid.

```
# Log linear model
house.lm2 <- lm(log(house_train$SalePrice)~., data=house_train)

summary(house.lm2)
```

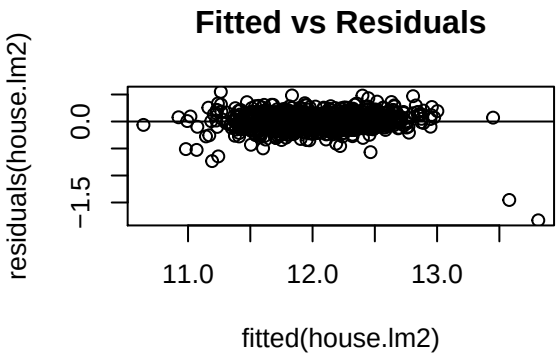
```
##
## Call:
## lm(formula = log(house_train$SalePrice) ~ ., data = house_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.82950 -0.07070 -0.00116  0.07625  0.54987
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)       4.001e+00  5.568e-01   7.186 1.26e-12 ***
## BldgType2fmCon    2.765e-02  3.308e-02   0.836 0.403458
## BldgTypeDuplex   -5.351e-02  2.903e-02  -1.844 0.065520 .
## BldgTypeTwnhs    -1.734e-01  2.949e-02  -5.882 5.42e-09 ***
## BldgTypeTwnhsE   -6.986e-02  1.940e-02  -3.600 0.000333 ***
## LotArea           2.115e-06  4.548e-07   4.651 3.71e-06 ***
## HouseStyle1.5Unf  1.062e-02  5.158e-02   0.206 0.836936
## HouseStyle1Story  5.639e-02  1.860e-02   3.031 0.002494 **
```

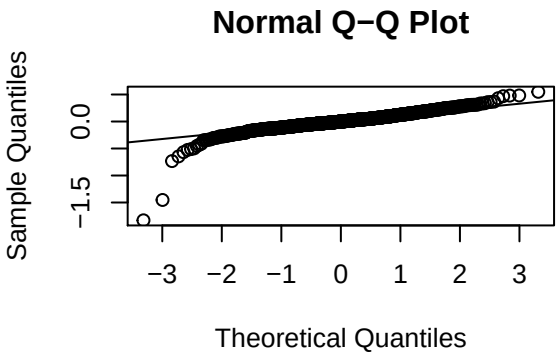
```
## HouseStyle2.5Fin -3.588e-02  6.224e-02  -0.577 0.564366
## HouseStyle2.5Unf  4.447e-02  5.472e-02   0.813 0.416580
## HouseStyle2Story -3.253e-02  1.906e-02  -1.707 0.088196 .
## HouseStyleSFoyer -1.119e-02  3.693e-02  -0.303 0.762037
## HouseStyleSLvl   -2.188e-03  2.750e-02  -0.080 0.936595
## OverallQual       1.010e-01  6.035e-03  16.742  < 2e-16 ***
## OverallCond       4.998e-02  5.047e-03   9.903  < 2e-16 ***
## YearBuilt         3.263e-03  2.883e-04  11.320  < 2e-16 ***
## BsmtFinType1BLQ   8.994e-04  1.907e-02   0.047 0.962387
## BsmtFinType1GLQ   1.519e-02  1.658e-02   0.916 0.359798
## BsmtFinType1LwQ  -3.184e-02  2.482e-02  -1.283 0.199819
## BsmtFinType1NO   -1.477e-01  3.499e-02  -4.221 2.64e-05 ***
## BsmtFinType1Rec  -2.094e-02  2.015e-02  -1.039 0.298895
## BsmtFinType1Unf  -6.015e-02  1.871e-02  -3.215 0.001345 **
## BsmtUnfSF        -2.698e-05  1.636e-05  -1.650 0.099299 .
## CentralAirY       1.161e-01  2.275e-02   5.105 3.92e-07 ***
## GrLivArea         2.799e-04  1.677e-05  16.695  < 2e-16 ***
## FullBath          2.934e-02  1.323e-02   2.217 0.026834 *
## BedroomAbvGr     -1.941e-03  8.058e-03  -0.241 0.809669
## GarageCars        7.285e-02  8.606e-03   8.465  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1527 on 1067 degrees of freedom
## Multiple R-squared:  0.8563, Adjusted R-squared:  0.8527
## F-statistic: 235.5 on 27 and 1067 DF,  p-value: < 2.2e-16
```

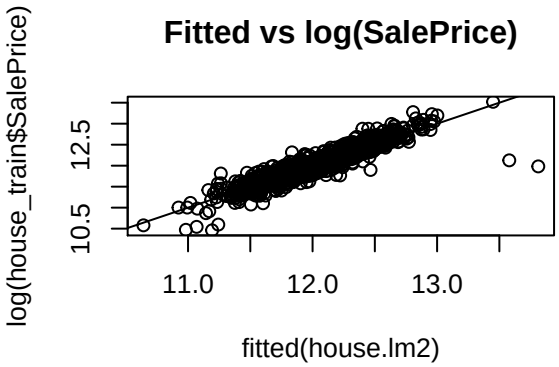
The R^2 has increased 6% up to 85.27%. This means our model is now able to explain 85.27% of the variability in the data. Interestingly, BsmtUnfSF (Unfinished basement sqaure feet), and BedroomAbvGr (Bedrooms above grade) are now the only variables which are not statistically significant to the model now. In the previous model, air conditioning and number of bathrooms were not statistically significant but now they are with the transformation. Unfinished basement square feet remains not significant and bedrooms above grade is a new addition to the variables that are not statistically significant.

Now we shall verify the model assumptions are valid with the log transformation.

```r
# Diagnostic plots
par(mfrow=c(2,2))
plot(fitted(house.lm2), residuals(house.lm2), main="Fitted vs Residuals")
abline(h=0)

qqnorm(residuals(house.lm2))
qqline(residuals(house.lm2))

plot(fitted(house.lm2), log(house_train$SalePrice), main="Fitted vs log(SalePrice)")
abline(0,1)
```







On the fitted vs residuals plot we see a random spread of points centered around 0 with no obvious pattern. The normal QQ plot looks much better than previous but the lower tail is still a potential cause for concern. The fitted vs log of sale price is much better and it shows a clear linear trend. All of these diagnostics together point to the model assumptions being valid.

Test

Here I will evaluate the performance of the model through its prediction accuracy using RMSE (Root Mean Squared Error).

```
#Calculate predictions
test <- house_test %>% select(-SalePrice)
pred.lm <- exp(predict(house.lm2, test))

#Calculate RMSE
RMSE.lm <- sqrt(sum((house_test$SalePrice-pred.lm)^2)/length(pred.lm))

kable(t(c(RMSE.lm, mean(house_train$SalePrice))), col.names = c("Linear Model RMSE", "Mean Sale Price"))
```

Linear Model RMSE	Mean Sale Price
29064.14	181043.1

The RMSE is quite large as a percentage of the mean sale price. This is a solid starting point but the linear model leaves much to be desired for prediction accuracy.

Generalized Linear Model

Generalized linear models are a great approach when your response variable has a distribution which comes from the exponential family. As noted earlier, the sale price of a house has a positively skewed distribution which closely matches a gamma distribution. For this reason I will be fitting a gamma GLM with a log link.

Train

```
# Gamma Generalized Linear Model With Log Link
house.glm = glm(house_train$SalePrice~., family=Gamma("log"), data=house_train)

summary(house.glm)
```

```
##
## Call:
## glm(formula = house_train$SalePrice ~ ., family = Gamma("log"),
##     data = house_train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.55874  -0.07715  -0.00690   0.06415   0.56829
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       3.730e+00  4.938e-01   7.555 9.01e-14 ***
## BldgType2fmCon    1.988e-02  2.934e-02   0.678 0.498159
## BldgTypeDuplex   -5.224e-02  2.574e-02  -2.029 0.042687 *
## BldgTypeTwnhs    -1.609e-01  2.615e-02  -6.151 1.09e-09 ***
## BldgTypeTwnhsE   -6.601e-02  1.721e-02  -3.836 0.000133 ***
## LotArea           2.466e-06  4.034e-07   6.114 1.36e-09 ***
## HouseStyle1.5Unf  2.581e-02  4.574e-02   0.564 0.572645
## HouseStyle1Story  7.072e-02  1.650e-02   4.286 1.98e-05 ***
## HouseStyle2.5Fin -8.855e-02  5.520e-02  -1.604 0.108958
## HouseStyle2.5Unf  2.856e-02  4.853e-02   0.588 0.556386
## HouseStyle2Story -3.307e-02  1.691e-02  -1.956 0.050734 .
## HouseStyleSFoyer  5.681e-03  3.275e-02   0.173 0.862342
## HouseStyleSLvl    8.321e-03  2.439e-02   0.341 0.733045
## OverallQual       9.658e-02  5.352e-03  18.045  < 2e-16 ***
## OverallCond       5.032e-02  4.476e-03  11.243  < 2e-16 ***
## YearBuilt         3.410e-03  2.557e-04  13.339  < 2e-16 ***
## BsmtFinType1BLQ   9.081e-04  1.691e-02   0.054 0.957183
## BsmtFinType1GLQ   2.035e-02  1.470e-02   1.384 0.166587
## BsmtFinType1LwQ  -3.216e-02  2.201e-02  -1.461 0.144254
## BsmtFinType1NO   -1.466e-01  3.103e-02  -4.726 2.60e-06 ***
## BsmtFinType1Rec  -2.032e-02  1.787e-02  -1.137 0.255851
## BsmtFinType1Unf  -4.721e-02  1.659e-02  -2.845 0.004524 **
## BsmtUnfSF        -3.236e-05  1.451e-05  -2.231 0.025912 *
## CentralAirY       9.538e-02  2.017e-02   4.728 2.57e-06 ***
## GrLivArea         3.445e-04  1.487e-05  23.170  < 2e-16 ***
## FullBath          1.242e-02  1.174e-02   1.058 0.290162
## BedroomAbvGr     -1.248e-02  7.146e-03  -1.747 0.080995 .
## GarageCars        6.119e-02  7.633e-03   8.017 2.82e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for Gamma family taken to be 0.01833503)
##
##     Null deviance: 175.528  on 1094  degrees of freedom
## Residual deviance:  22.245  on 1067  degrees of freedom
## AIC: 25240
##
## Number of Fisher Scoring iterations: 6
```

Here we see that the number of full bathrooms and the number of bedrooms above grade are statistically insignificant to the model.

Let us perform an asymptotic chi-squared test to determine if our model has a good fit to the data.

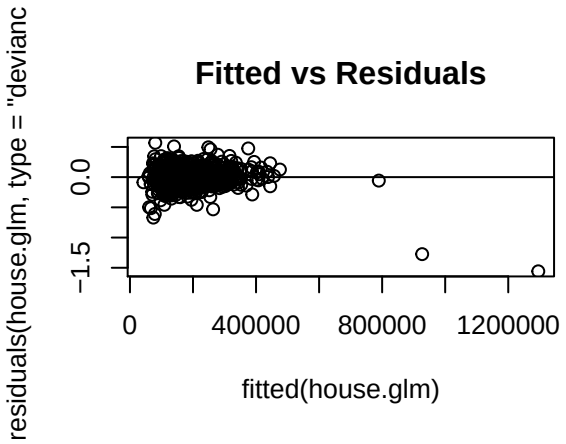
```
# Likelihood Ratio Test
kable(t(c(LRT=house.glm$deviance,
  df=house.glm$df.residual,
  p.val=pchisq(house.glm$deviance, house.glm$df.residual,lower=F))))
```

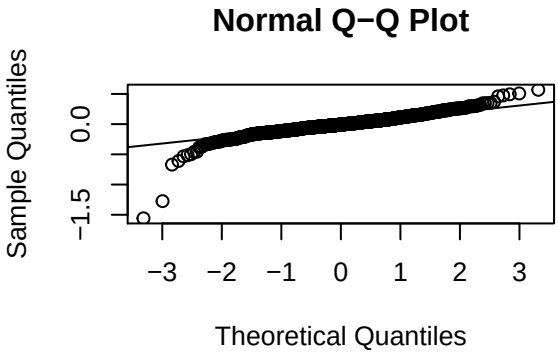
LRT	df	p.val
22.24508	1067	1

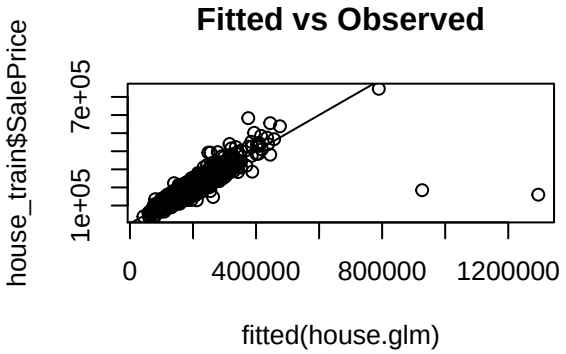
Here we see that the p-value is 1 so we fail to reject the null hypothesis and conclude that the proposed model fits the data well in comparison to the saturated model.

Now let us examine the diagnostic plots to determine if the statistical inference is valid

```
# Diagnostic plots
par(mfrow=c(2,2))
plot(fitted(house.glm), residuals(house.glm, type="deviance"), main="Fitted vs Residuals")
abline(h=0)

qqnorm(residuals(house.glm, type="deviance"))
qqline(residuals(house.glm, type="deviance"))

plot(fitted(house.glm), house_train$SalePrice, main="Fitted vs Observed")
abline(0, 1)
```







In the fitted vs residuals plot we see a mostly random scatter of points, however there does appear to be a slight pattern in the residuals which is not ideal. The normal QQ plot looks great in center but at the lower tail there is clear deviation from normality which is not great. Lastly, the fitted vs observed plot appears mostly linear. Based on these diagnostics I would say that we can moderately trust the asymptotic inference but its certainly not great.

Test

Here I will evaluate the performance of the model through its prediction accuracy using RMSE (Root Mean Squared Error).

```
#Calculate predictions
test <- house_test %>% select(-SalePrice)
pred.glm <- exp(predict(house.glm, test))

#Calculate RMSE
RMSE.glm <- sqrt(sum((house_test$SalePrice-pred.glm)^2)/length(pred.glm))

kable(t(c(RMSE.glm, mean(house_train$SalePrice))), col.names = c("GLM RMSE", "Mean Sale Price"))
```

GLM RMSE	Mean Sale Price
27171.5	181043.1

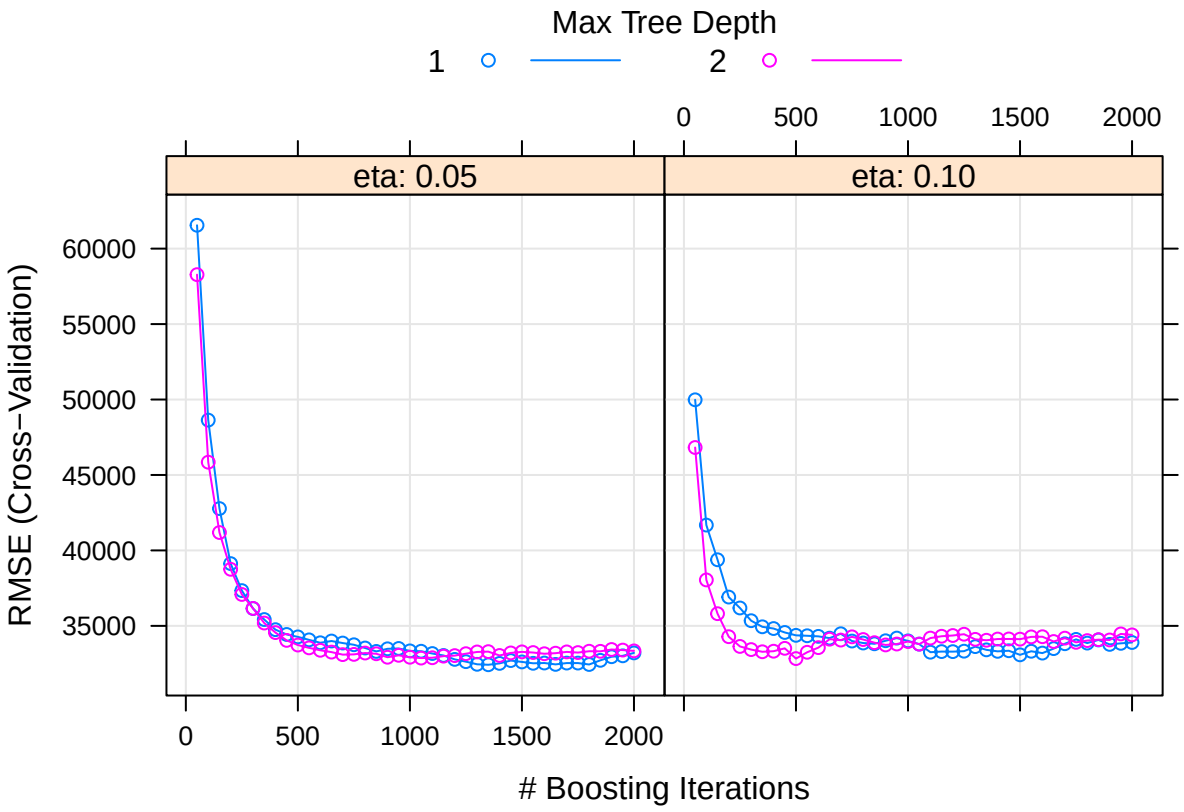
Here we see a slight improvement in RMSE as compared to the linear regression but it is still not ideal.

Boosted Random Forest

A boosted random forest model is a great ensemble machine learning algorithm that can perform decently for all problems. Here I will be using 5-fold cross vaildation and grid search for hyperparameter optimization.

Train

```
set.seed(1000)
model <- SalePrice~.

control <- trainControl(method = "cv", number=5, classProbs=F)

tune_grid <- expand.grid(nrounds = seq(from=50, to=2000, by=50),
                         eta = c(0.05, 0.1),
                         max_depth=c(1,2),
                         gamma=0,
                         colsample_bytree=c(0.1),
                         min_child_weight=1,
                         subsample=c(0.05))


house.xgb <- caret::train(model,
                                data=house_train,
                                method="xgbTree",
                                tuneGrid=tune_grid,
                                trControl=control)

plot(house.xgb)
```




```
house.xgb$bestTune
```

```
##    nrounds max_depth  eta gamma colsample_bytree min_child_weight subsample
## 27    1350         1 0.05     0              0.1                1      0.05
```

```
varImp(house.xgb)
```

```
## xgbTree variable importance
##
##   only 20 most important variables shown (out of 27)
##
##                  Overall
## GrLivArea        100.000
## OverallQual       97.475
## GarageCars        59.395
## YearBuilt         57.240
## LotArea           47.623
## FullBath          32.879
## BsmtUnfSF         28.420
## BedroomAbvGr      15.902
## BsmtFinType1GLQ   12.006
## OverallCond       10.007
## BsmtFinType1Unf    5.611
## HouseStyle2Story   5.465
## CentralAirY        4.849
## BldgTypeTwnhsE     3.058
## BsmtFinType1LwQ    2.596
```

```
## BldgTypeDuplex     2.448
## HouseStyle1Story   2.418
## BldgType2fmCon     1.433
## BsmtFinType1NO     1.318
## BsmtFinType1BLQ    1.195
```

The plot here shows the performance of the model as it is trained over the grid search. We also have a list of the most important variables to the model. We see that ground living area and overall quality are rated as extremely important to the model's success.

Now I will move on to testing the model.

Test

```
#Calculate test set predictions
test <- house_test %>% select(-SalePrice)
pred.xgb <- predict(house.xgb, test)

#Calculate test set RMSE
RMSE.xgb <- sqrt(sum((house_test$SalePrice-pred.xgb)^2)/length(pred.xgb))

#Calculate train set predictions
test2 <- house_train %>% select(-SalePrice)
pred.xgb2 <- predict(house.xgb, test2)

#Calculate train set RMSE
RMSE.xgb2 <- sqrt(sum((house_train$SalePrice-pred.xgb2)^2)/length(pred.xgb2))

#Combine RMSE into single matrix
RMSE_matrix.xgb <- rbind(c(RMSE.xgb, mean(house_train$SalePrice)), c(RMSE.xgb2, mean(house_train$SalePr

rownames(RMSE_matrix.xgb) <- c("Test", "Train")

kable(RMSE_matrix.xgb,
      col.names = c("XGBoost RMSE", "Mean Sale Price"))
```

	XGBoost RMSE	Mean Sale Price
Test	32311.77	181043.1
Train	29840.95	181043.1

Here we see that the testing RMSE is higher than the two previous models which is not great given the added complexity of interpreting the model. Our train and test error here are fairly close together but there could be an argument for some overfitting over the model.

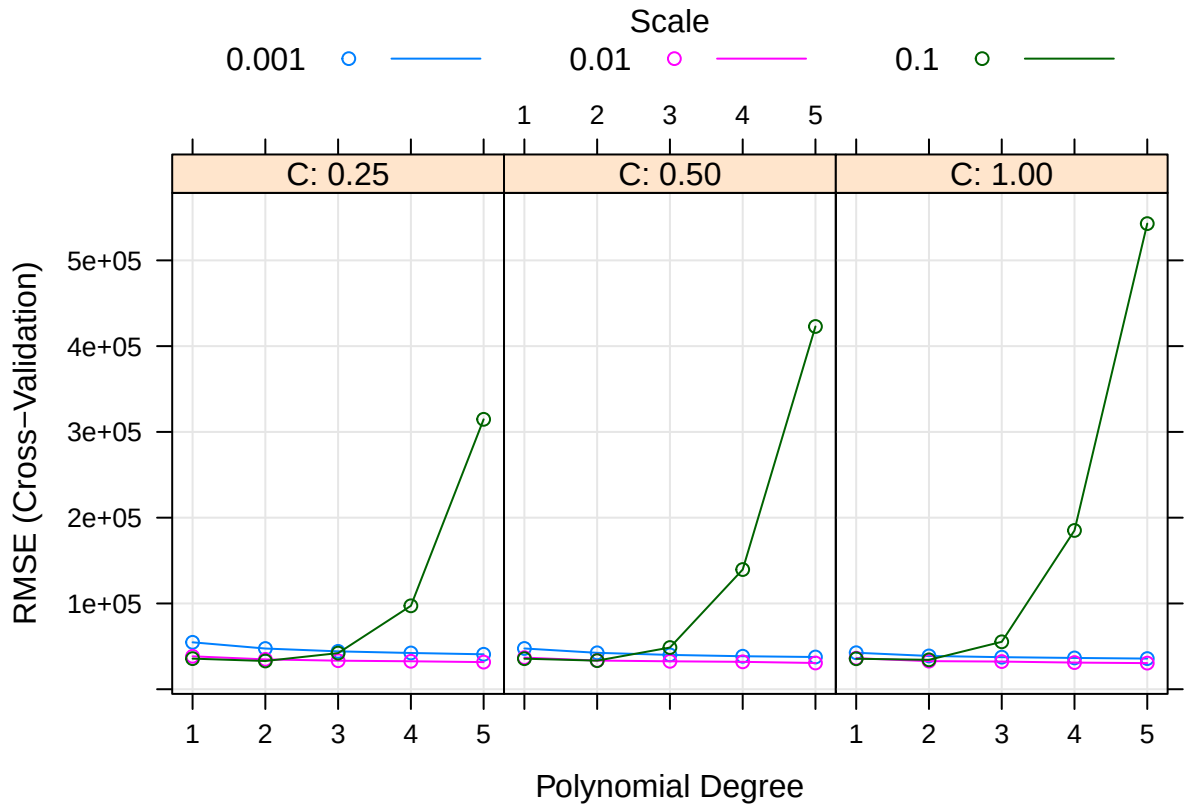
Support Vector Machine

Support vector machines are another great general purpose machine learning algorithm that can be applied to numerous problems. Here I will be using 5-fold cross validation with a polynomial kernel function.

Train

```
set.seed(1000)
model <- SalePrice~.
```

```
control.svm <- trainControl(method = "cv", number=5, classProbs=F)

tune_grid.svm <- expand.grid(degree=c(1,2,3,4,5),
                             scale=c(0.001,0.01,0.1),
                             C=c(0.25,0.5,1))

house.svm <- caret::train(model,
                          data=house_train,
                          method="svmPoly",
                          tuneGrid=tune_grid.svm,
                          trControl=control.svm)

plot(house.svm)
```




```
house.svm$bestTune
```

```
##    degree scale C
## 42      5  0.01 1
```

The plot here shows the performance of the model as it is trained over the grid search.

Test

Here I will calculate the test RMSE and the train RMSE.

```
#Calculate test set predictions
test <- house_test %>% select(-SalePrice)
```

```
pred.svm <- predict(house.svm, test)

#Calculate test set RMSE
RMSE.svm <- sqrt(sum((house_test$SalePrice-pred.svm)^2)/length(pred.svm))

#Calculate train set predictions
test2 <- house_train %>% select(-SalePrice)
pred.svm2 <- predict(house.svm, test2)

#Calculate train set RMSE
RMSE.svm2 <- sqrt(sum((house_train$SalePrice-pred.svm2)^2)/length(pred.svm2))

#Combine RMSE into single matrix
RMSE_matrix.svm <- rbind(c(RMSE.svm, mean(house_train$SalePrice)), c(RMSE.svm2, mean(house_train$SalePr

rownames(RMSE_matrix.svm) <- c("Test", "Train")

kable(RMSE_matrix.svm,
      col.names = c("SVM RMSE", "Mean Sale Price"))
```

	SVM RMSE	Mean Sale Price
Test	26460.45	181043.1
Train	23436.55	181043.1

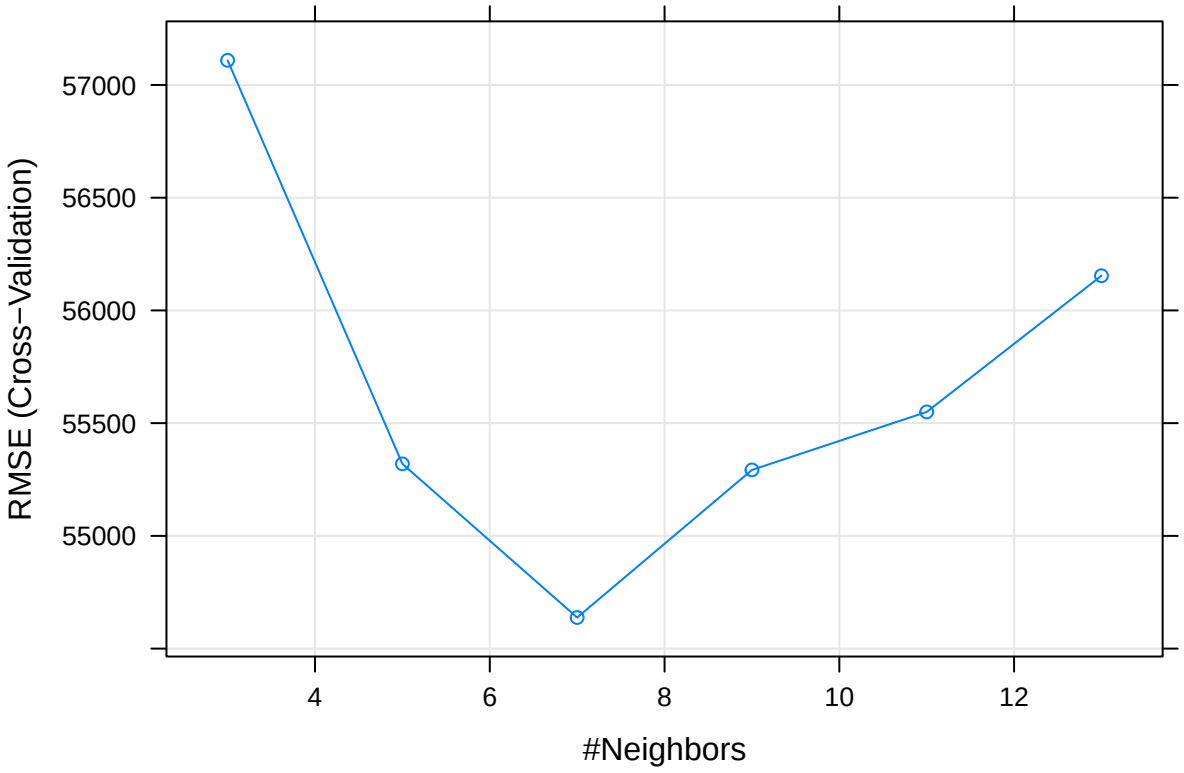
This RMSE is the best of all of the models so far. There is a bit of a difference between the train and the test error so there is an argument to be made for overfitting in this model.

K-Nearest Neighbors

Finally, I will be applying the KNN algorithm to this problem. KNN is a very simple machine learning algorithm that I would not expect to have great performance in this instance. However, it is always a good idea to try out multiple models.

Train

```
set.seed(1000)
model <- SalePrice~.

control.knn <- trainControl(method = "cv", number=5, classProbs=F)

tune_grid.knn <- expand.grid(k=c(3,5,7,9,11,13))

house.knn <- caret::train(model,
                          data=house_train,
                          method="knn",
                          tuneGrid=tune_grid.knn,
                          trControl=control.knn)

plot(house.knn)
```




```
house.knn$bestTune
```

```
##   k
## 3 7
```

Here we see the plot of the model performance versus the number of neighbors used for the regression.

Test

```
#Calculate test set predictions
test <- house_test %>% select(-SalePrice)
pred.knn <- predict(house.knn, test)

#Calculate test set RMSE
RMSE.knn <- sqrt(sum((house_test$SalePrice-pred.knn)^2)/length(pred.knn))

#Calculate train set predictions
test2 <- house_train %>% select(-SalePrice)
pred.knn2 <- predict(house.knn, test2)

#Calculate train set RMSE
RMSE.knn2 <- sqrt(sum((house_train$SalePrice-pred.knn2)^2)/length(pred.knn2))

#Combine RMSE into single matrix
RMSE_matrix.knn <- rbind(c(RMSE.knn, mean(house_train$SalePrice)), c(RMSE.knn2, mean(house_train$SalePr

rownames(RMSE_matrix.knn) <- c("Test", "Train")
```

```
kable(RMSE_matrix.knn,
      col.names = c("KNN RMSE", "Mean Sale Price"))
```

	KNN RMSE	Mean Sale Price
Test	61152.15	181043.1
Train	47153.99	181043.1

Here we see very poor performance on the test set and the worst model performance overall by a large margin. Additionally we certainly have overfitting issues given the discrepency between the train and the testing error. Increasing the value of k may be able to remedy the overfitting, however, this model does not appear to be appropriate for this problem at all.

Conclusion

Of all of the models presented in this project, I believe that the gamma Generalized Linear Model is the best approach. The gamma GLM had the second best prediction RMSE while also having the benefit of vastly superior interpretability to the SVM. If the goal is purely prediction accuracy then it appears the SVM would be the best approach here.

Looking back on this project I believe that many improvements can be made to build better models and to increase prediction accuracy. Here is a list of topics which I believe could play a role in the improvement of this project in the future:

- More rigorous feature selection
- Outlier removal
- Model stacking

Furthermore, I believe that there is great potential for future analysis of this dataset which includes:

- Clustering models
- Analysis on different response variables

Ultimately, the analysis of this data set has proved to be very insightful and I believe that if the improvements I have proposed are enacted, then even greater insights will follow, and prediction accuracy will increase greatly.